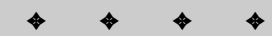
CHAPTER 31

Adding Fonts



In This Chapter

Introducing fonts

Fonts on the Web

Font selection considerations

Controlling font selection

Choosing font sizes

Using condensed and expanded fonts

Adding small caps and other text decorations

CSS gives you incredible flexibility in defining fonts for your pages. You can specify multiple font faces, sizes for each element, the width between letters, and the spacing between words. You can even add interesting typographical effects, such as all caps on the first line (which many magazines and newspapers use), and a first letter that spans multiple lines. This chapter explains how to create all these effects using CSS.

Introducing Fonts

Most people think of fonts as letters. Fonts are really the representation of letters with *glyphs* (pictures). The letters a, b, and c can look very different in Arial, Times New Roman, and Lucida Console. Even though the letters are the same, the glyphs are different.

Font families

The two major font families are *serif* and *sans serif*. Serif fonts include little decorations on the edges of letters. The most common serif font is Times Roman or Times New Roman, depending on your platform. Sans serif (meaning without serif) fonts are much plainer. As you can see in Figure 31-1, Arial and Lucida Console are sans serif fonts.

When you send text over the Internet, normally you send only your choices of the letters that make up the text. The recipient's browser renders your text by using whatever font face and size its owner has configured it to use. Until recently, you couldn't send the choice of font — and you certainly couldn't send the actual font. With CSS, Web authors have increasing control over font choice.

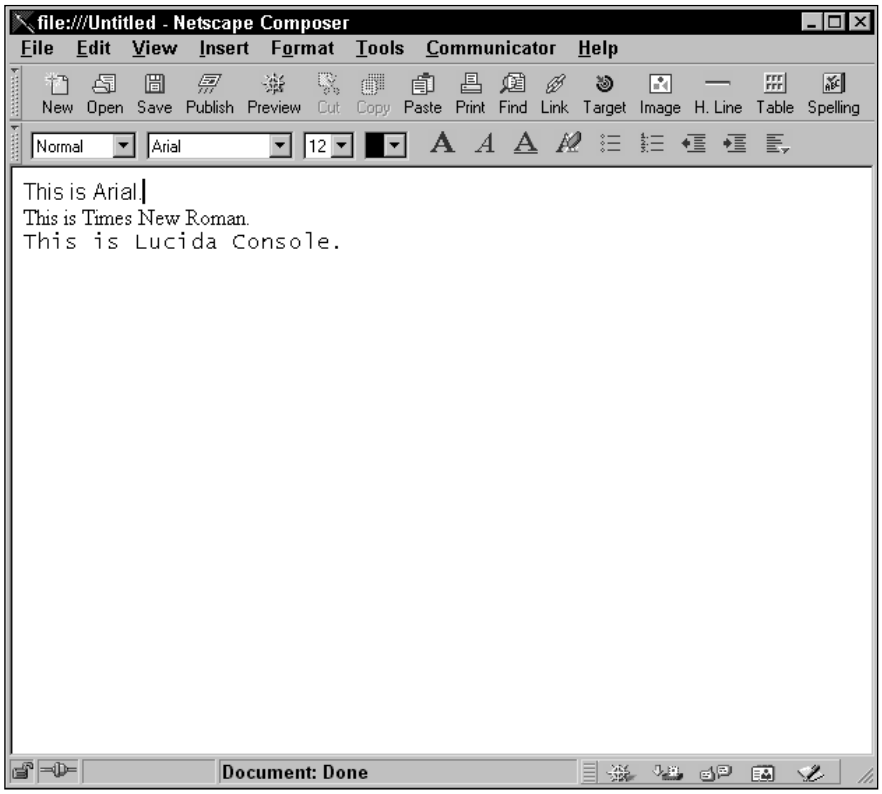

file:///Untitled - Netscape Composer
File Edit View Insert Format Tools Communicator Help
New Open Save Publish Preview Cut Copy Paste Print Find Link Target Image H. Line Table Spelling
Normal
Arial
12
This is Arial.
This is Times New Roman.
This is Lucida Console.
Document: Done

Figure 31-1: Sample fonts displayed in Netscape Communicator 4.7

One reason text renders so quickly in your browser is that nothing more than choices of letters — not the actual glyphs — are sent to your computer. With CSS, the Web developer now has four possible choices:

- ✦ Send text without specifying a font
- ✦ Send text that specifies which font the browser should use (usually as a list of choices in case the first choice isn't present)
- ✦ Rely on CSS2's support for intelligent matching or font synthesis, if the specified font is unavailable on the target computer — if supported by the browser
- ✦ Send the text with the URL of a *downloadable* font (which requires specialized software)

Fonts versus the image of fonts

Most Web developers who want to include fonts other than those commonly available create their text as graphics and send graphics instead of text. This approach can substantially slow down the load time of their pages. On the upside, it does guarantee that anyone with a graphics-capable browser will see the font the same way. If you want to show a small amount of a foreign language, using an image of text is probably the best way to go; your visitors need not load a special font set or run any particular operating system.

Using Local Fonts

If you decided you don't want to be stuck using the default font on the system — and you don't want to play games with creating images of text instead of sending the text — then you may want to take advantage of the CSS capability that specifies which local fonts the system ought to use. Because you can't know which fonts are installed on the browser — and you can't even know whether the browser is a PC, a Mac, or something else (unless you are developing for an intranet) — CSS gives you the option of specifying a list of fonts from which the browser can select. It will use the first listed font that matches one currently installed. To be even more thorough, after specifying each specific font name you would prefer the browser to use, you can specify font family names; doing so ensures that if none of your specified fonts is present, the browser selects something that at least looks similar.

Font-Selection Considerations

There are two major factors to consider when selecting the fonts for your page. The first relates to aesthetics:

- ✦ Do the fonts on the page work together?
- ✦ How many different fonts work on one page?
- ✦ Should you mix serif and sans serif fonts?

The second factor relates directly to whether the browser can re-create the page as you initially conceived it.

The aesthetics of font selection

Different fonts connote different qualities. Do you want the artistic sweep of a cursive font such as Brush Script? Do you want the clean lines of a sans serif font such as Helvetica? There are good reasons to use just about any font, but you'll want to make sure the font you select helps convey your message, rather than getting in the way of it.

Consider whether to use the same font for headings as you do for paragraph text. Serif fonts such as Times Roman (which have little hooks on the ends of the letters) are easier to read. Sans serif fonts such as Helvetica are cleaner looking and leave more white space. For these reasons, many people choose to use serif fonts for paragraph text and sans serif text for headings. You can use more than one font family in your document, but normally it's wise to avoid using more than two font families in one document. Too many font families can contribute to a cluttered or disorganized look.

Availability of local fonts

Even if you can create the perfect page, with just the right fonts, on your local system, what can you do to ensure that the browser renders your page as you would like it to be rendered? You can send a list of fonts you would like the browser to use, and the browser will use the first one in your list that it has. You aren't limited in how many you can list. If the default browser font would be noticeably worse, you would be well advised to list every font that would work well.

Controlling Font Selection

You can use the font-family property to indicate which font you want the browser to use. You can identify specific fonts, font families (serif, sans serif, cursive, fantasy, monospace), or both. If you list both, include the font families at the end so the browser will look for the most precise match before it uses a generic font family.

```
P {
        font-family: "Bookman Antiqua", Georgia, "Times New
Roman", "Times Roman", Times, serif;
}
```

The rule just given tells the browser the paragraph text should be rendered with Bookman Antiqua font; if that is not available, Georgia; if Georgia is not available, Times New Roman; if Times New Roman is not available, Times Roman; if Times Roman is not available, Times — or, if none of those fonts are available, any serif font. Note, any font name with more than one word in it should be contained in double quotes.

Internet Explorer does a pretty good job of handling font families. It does provide a script font when you specify cursive, but it gives you the same font for fantasy as for sans serif. Netscape gives you sans serif for both cursive and fantasy font families. However, if you specify "Comic Sans MS" font (which is part of the fantasy family), it will work on both Netscape and Internet Explorer. The moral of the story: Specify the exact font you want used; don't rely on font families.

Choosing Font Sizes

You can set the font size with the font-size property. Setting the font size is useful because you can be sure the browser will lay out headings as you want them to appear. Normally, H6 elements render the same as regular paragraph text. You can change this and give greater importance to H6 elements by increasing the font size, changing the font style, or increasing the line height.

The font-size property takes a value in points (pt):

```
H6 {
        font-size: 14pt;
}
```

Using Condensed and Expanded Fonts

You can condense and expand fonts both horizontally — by compressing or spreading out the letters or the words — and vertically — by decreasing or increasing the white space between lines, using CSS. Table 31-1 shows each type of property at work.

Table 31-1
Expanding and Condensing Lines, Words, and Letter Spacing

Type of Spacing	*CSS*	*Example*
Expanding letter spacing	`P.wide {letter-spacing: .5pt; }`	This is an example of increased letter spacing.
Condensing letter spacing	`P.narrow {letter-spacing: -.8pt; }`	This is an example of decreased letter spacing.

Continued

Table 31-1 *(continued)*

Type of Spacing	*CSS*	*Example*
Condensing space between rows	`P.crowded { line-height: 10pt; }`	This is an example of decreased line height; notice the lines really crowd each other.
Expanding space between rows	`P.speaking-slowly { line-height: 18pt; }`	This is an example of increased line height; notice the lines are a bit more spread out.
Expanding space between words	`P.speaking-slowly { word-spacing: 1em; }`	This is an example of expanded word spacing.

Condensing and expanding horizontally

To condense or expand fonts horizontally (meaning the letters appear closer to each other than they normally would), use the letter-spacing property to add to the default value of space between letters.

```
H1 {
      letter-spacing: .2em;
}
```

In the preceding example, the letter spacing will be increased by 0.2em for `H1` elements.

```
P.legal {
      letter-spacing: -.2em;
}
```

In the preceding example, the letter spacing will be decreased by 0.2em for `P` elements with a class of *legal*.

You can also expand or condense fonts by increasing or decreasing the white space between words. The property for this is called *word spacing*. It works like letter spacing, which adds to the default amount of white space between words.

```
P.spread-out {
      letter-spacing: .2em;
      word-spacing: .5em;
}
```

The previous example creates a class called *spread out,* in which there is an extra 0.2em of space between letters and an extra 0.5em of space between words.

If you spread out letters, you probably want to spread out words as well, or it might not be clear where one word ends and the next one begins.

Condensing and expanding vertically

Condensing or expanding text on the page vertically — that is, leaving more or less white space between lines — is easily accomplished with the line-height property. By default, line height is two pixels greater than font-size. For example, if you have a 12pt font for your paragraph text, the browser will automatically render the text with 14pt line height — 12 points for the letter and one blank point to provide blank space at both the top and bottom of the letter. If you want to condense the line height, you can do so by specifying a line height less than 2 points greater than the font size. If you want to increase the white space between lines, increase the line height to more than two points greater than the font size.

```
P {
        font-size: 12pt;
        line-height: 14pt;
}
P.double {
        font-size: 12pt;
        line-height: 24pt;
}
P.smooshed {
        font-size: 12pt;
        line-height: 12pt;
}
```

For clarity in this example, the font size is repeated for each class, but it needn't be repeated. The double class double-spaces the 12pt text. The smooshed class leaves no white space between lines.

Adding Small Caps and Other Decorations

You can do two very cool things with CSS. They take advantage of the pseudo-elements called `first-line` and `first-letter`. You learned about pseudo-classes in Chapter 26; pseudo-classes are related to the `A` element. The descriptors for these classes (`A:link`, `A:visited`, `A:active`) can be assigned their own values to further customize the look of linked text.

Changing the first line

Pseudo-elements relate to the typographical layout of your page. The property called first-line enables you to simulate newspapers and magazines, which often apply a different typographical effect to the first line of text in a paragraph (all caps, for example).

```
P.first:first-line {
       font-style: small-caps;
}
```

When you have this rule in your style sheet, you can use the following HTML:

```
<P class="first">In Chicago today, it was announced that people
with more money live in nicer neighborhoods.</P>
```

This approach would render text similar to the following example in your browser:

IN CHICAGO TODAY, IT WAS ANNOUNCED
that people with more money live in
nicer neighborhoods.

Changing the first letter

The property called first-letter enables you to create text with a first letter that extends below the first line. For example, in many magazines, the first letter of the first paragraph in an article is two lines high.

```
P.drop-cap:first-letter {
       font-size: 36pt;
       float: left;
}
```

With this rule in your style sheet, you can use the following HTML:

```
<P class="drop-cap">In Chicago today, it was announced that
people with more money live in nicer neighborhoods.</P>
```

In a browser that supports this feature (which currently excludes both Netscape 4 and Internet Explorer 4), the code just given renders like this:

In Chicago today, it was announced
that people with more money live
in nicer neighborhoods.

From Here

Proceed to Chapter 32 and explore the essentials of Web page design.

Refer back to Chapter 22 to review frames.

Summary

In this chapter, you learned about specifying the fonts you want used with your Web page. You also learned that fonts are really pictures of letters called *glyphs*. This chapter discussed the factors you should consider when selecting fonts and the number of font faces you probably don't want to exceed. You learned to specify font size and font spacing (both horizontally and vertically), and you learned to add very cool text decorations, such as all caps on the first line, or a first letter that spans several rows.

✦ ✦ ✦